No. 14-1351

UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT

ORACLE AMERICA, INC.

Appellant,

v.

GOOGLE, INC.

Appellee.

Appeal from the United States Patent and Trademark Office,
Patent Trial and Appeal Board,
Reexamination Control No. 95/001,548.

BRIEF OF APPELLEE

Scott T. Weingaertner
KING & SPALDING LLP
1185 Avenue of the Americas
New York, NY 10036
Telephone: (212) 556-2227
Facsimile: (212) 556-2222
sweingaertner@kslaw.com

Robert T. Neufeld
KING & SPALDING LLP
1180 Peachtree St., NE
Atlanta, GA 30309
Telephone: (404) 572-3505
Facsimile: (404) 572-4600
bneufeld@kslaw.com

Daryl L. Joseffer
Counsel of Record
KING & SPALDING LLP
1700 Pennsylvania Ave. NW
Washington, DC 20006
Telephone: (202) 737-0500
Facsimile: (202) 626-3737
djoseffer@kslaw.com

Adam M. Conrad
KING & SPALDING LLP
100 N. Tryon St., Ste. 3900
Charlotte, NC 28202
Telephone: (704) 503-2600
Facsimile: (704) 503-2622
aconrad@kslaw.com

Counsel for Appellee Google Inc.

July 28, 2014 *additional counsel listed on inside cover

Brian C. Banner
KING & SPALDING LLP
401 Congress Ave., Ste. 3200
Austin, TX 78701
Telephone: (512) 457-2000
Facsimile: (512) 457-2100
bbanner@kslaw.com

CERTIFICATE OF INTEREST

Counsel for Appellee Google Inc. certifies the following:

1. The full name of every party represented by me is Google Inc.

2. The name of the real party in interest represented by me is Google Inc.

3. Google Inc. is a publicly traded company (NASDAQ: GOOG) and no publicly held companies own 10% or more of Google Inc.'s stock.

4. The names of all firms and the partners or associates that appeared for the parties now represented by me in the trial court or are expected to appear in this Court are:

King & Spalding LLP: Brian C. Banner; Adam M. Conrad; Daryl L. Joseffer; Robert T. Neufeld; Scott T. Weingaertner

This 28th day of July 2014.

/s/ Daryl L. Joseffer
Daryl L. Joseffer

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Cases

Statutes

Other Authorities

STATEMENT OF RELATED CASES

There is no related case under Fed. Cir. R. 47.5. This Court recently decided an unrelated copyright case involving the same parties. *See Oracle America, Inc. v. Google Inc.*, 750 F.3d 1339 (Nos. 13-1021, -1022) (Fed. Cir. May 9, 2014) (O'Malley (writing), Plager, Taranto, J.J.). Although Appellant Oracle America, Inc. ("Oracle") initially asserted a number of patents in that district court action, including the patent at issue here (U.S. Patent No. 6,910,205), Oracle voluntarily dismissed with prejudice all claims related to the '205 patent and did not appeal any issues related to any of the patents. The outcome of this appeal, which concerns the validity of the '205 patent in light of a prior art reference, will therefore not affect the other case.

STATEMENT OF JURISDICTION

Appellee Google Inc. ("Google") agrees with the jurisdictional statement provided by Oracle.

INTRODUCTION

The technology at issue in this *inter partes* reexamination relates to computer software called interpreters (including, for example, virtual machines). *See* A86 1:54-58. An interpreter provides one means of converting instructions in computer programs into code that can be executed by a computer. Interpreters promote flexibility in computer programming. Rather than write separate programs for a multitude of computer architectures, a programmer can design a single program that the interpreter can understand and decode for any machine on which the interpreter runs. The drawback is that decoding takes time. Interpreted programs are often slow compared to programs written for a specific computer and compiled into code that can be directly executed by that machine.

The disputed patent (U.S. Patent No. 6,910,205) purports to improve the speed of interpreters by converting "frequently used code sequences" into code that a computer can execute directly rather than in an "interpreted fashion." A91 11:16-19. Put another way, the disclosed method replaces a set of virtual machine instructions for an interpreter with a "hybrid" set of instructions that contains both virtual machine instructions and directly executable "native" machine instructions.

The Board determined that this solution was not novel. It rejected claims 1-4, 8, 15, 16, and 18-21 as anticipated by Peter Magnusson's article entitled *Partial Translation*, Swedish Institute of Computer Science Technical Report (T93.5)

(Oct. 1993) ("Magnusson").[1] Magnusson disclosed an improved interpreter for the purpose of simulating computer programs designed for one type of computer on "a dissimilar machine." A912. Four years prior to the '205 patent, Magnusson had noted the performance "slowdown" of interpreters and proposed a "hybrid approach" to "combine the performance benefits of direct execution (running generated native code) with the flexibility and accuracy of interpretation." A917, A912.

The Board's decision is well supported. In addition to the text and figures in Magnusson itself, Google provided two expert declarations demonstrating that Magnusson discloses to one of ordinary skill in the art each and every element of the challenged claims. *See* A891-99, A1063-66. The Examiner credited Google's expert and adopted Google's reasoning virtually across the board. *See* A21-44. The similarities between Magnusson and the '205 patent are so clear that Oracle no longer contests the Board's conclusion that Magnusson discloses each and every element of claims 1 and 8 of the '205 patent. *See* Oracle Br. 2 (issues 1 and 2).

On appeal, Oracle challenges the Board's decision largely by mischaracterizing it. Oracle bases its claim construction argument (for some but not all claims) on the notion that the Board construed the term "overwriting" to

---

[1] The Magnusson reference is included in the record twice: first at A479-95 as part of the reexamination request, then again at A910-26 as an attachment to Google's expert's declaration. This brief cites to the latter copy because Google's expert referred to hand-written page numbers on this copy.

mean "*not* writing over." Oracle Br. 23 (emphasis in original). That is simply not true. The Board construed "overwriting" to mean "replacing some information in a computer file with new information, rather than literally writing over an existing information." A8. In other words, "overwriting" is broader than "literally writing over." When read correctly, rather than in the narrow sense now advanced by Oracle, the Board's construction is consistent with the term's ordinary meaning as well as the written description, particularly in view of the "broadest reasonable construction" standard.

Oracle's enablement argument fares no better, turning on alleged typographical errors and the absence of minor details. The Examiner put it well: Oracle implicitly contends that one of ordinary skill "would have required a disclosure of error-free code examples, explicit details on how to add a new instruction to existing code, and drawings that provide the exact same detail as the accompanying text in order to avoid undue experimentation." A41. The Examiner and the Board properly rejected that cramped view of the law.

STATEMENT OF THE ISSUES

The questions presented are:

(1) Whether substantial evidence supports the Board's finding that Magnusson anticipated claims 1-4, 8, 15, 16, and 18-21 of the '205 patent.

(2) Whether substantial evidence supports the Board's finding that Magnusson is an enabling reference.

STATEMENT OF THE CASE

This appeal arises from an *inter partes* reexamination of the '205 patent, assigned to Oracle. Google requested the reexamination in February 2011, *see* A100, and the PTO found substantial new questions of patentability, *see* A762, A771, A774-75. As relevant here, the Examiner rejected claims 1-4, 8, 15, 16, and 18-21 as anticipated by Magnusson under 35 U.S.C. § 102. The Board "sustain[ed] the Examiner's rejections of" all challenged claims. A12.

A. The '205 Patent

The claimed technology relates to "interpreters"—software programs that convert source code instructions into code that can be executed by a computer. *See* A86 1:10-2:32. Typically, interpreted programs contain code that is "not designed for any specific microprocessor or computer architecture." A88 5:23-25; *see also* A87 4:10-12. One well-known interpreter is the Java virtual machine. A86 1:45-47. Some interpreted programs are written in the form of "virtual machine instructions" that are understood by the virtual machine, rather than any specific computer. This approach promotes "flexibility" in computer programming because "the virtual machine instructions may be run, unmodified, on any

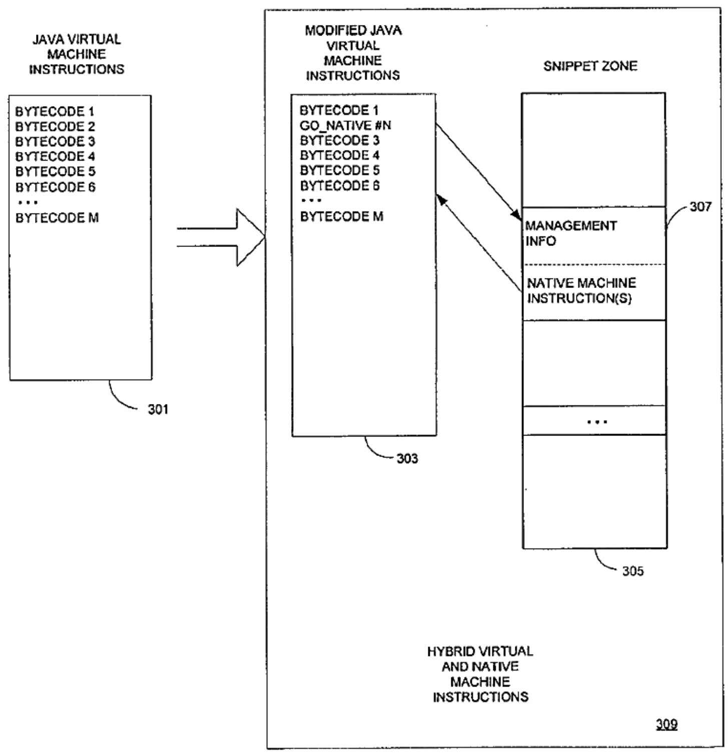
computer system that has a virtual machine implementation," making the program "portable" from computer to computer. A86 1:47-51.

Flexibility comes at the expense of execution speed because computers do not directly execute virtual machine instructions, which must first be interpreted. Computers can directly execute *native* machine instructions—that is, programs that are compiled into code "designed for a specific microprocessor or computer architecture." A87 4:13-15. The "native machine or computer system" can execute compiled, native code "without necessitating decoding." A86 1:66-67. Interpreted programs require the additional step of converting the virtual machine instructions into native machine code. To do this, the virtual machine "decode[s] each instruction before it is executed," A86 1:60-67, 45-47, 55-58, and then "jumps to the piece of native machine code that implements" the virtual instruction, A88 5:46-47. Execution of interpreted programs is therefore "typically quite slower" than execution of programs that have been compiled into native code. A86 1:63-64; *see also* A88 5:61-63.

The '205 patent aims to "increas[e] the execution speed of interpreters" by replacing certain virtual machine instructions with "a hybrid of virtual and native machine instructions." A86 1:10-13; *see also id.* at 2:27-29. This hybrid approach may expedite "[t]he execution of an interpreted program" by executing "frequently used code sequences . . . in native code rather than an interpreted fashion." A91

11:16-19. The hybrid instructions essentially enable the virtual machine to avoid the time-consuming decoding step for some instructions by executing native machine instructions directly. *See* A88 6:23-45.

Figure 5 depicts one embodiment of the claimed method. *See* A77; *see also* A87 3:46-48. In this figure, the claimed method generates “hybrid virtual and native machine instructions” from a set of original virtual machine instructions (designated 301 on the left-hand side). A89 7:63-64. The hybrid instructions include “a new virtual machine instruction” labeled “go_native.” A89 7:23-26. The go_native instruction “replaces or overwrites the initial virtual machine instruction of the selected portion of the function” (here, bytecode 2). A89 7:26-28; *see also id.* at 8:2-5. Upon executing the go_native instruction, the interpreter finds “a pointer” to a “snippet” (designated 305 on the right-hand side) that includes native machine instructions that “perform the same operations” as the virtual machine instructions labeled bytecodes 2-5. A89 8:31-32; *see also* A90 10:51-54. Executing the snippet



eliminates the need to interpret each bytecode separately. After executing the native machine instructions, "the interpreter continues with the execution of" the next virtual machine instruction (bytecode 6) "as if no snippet existed." A89 8:31-34.

The virtual machine may also "reverse the introduction of snippets" by restoring the original virtual machine instruction. A90 10:55-62. In the context of Figure 5, the claimed method stores "the original bytecode 2 which was overwritten by the go_native bytecode . . . so that the original bytecode sequence may be restored when the snippet is removed." A89 8:11-16. "The address of the original virtual machine instruction" is stored and "acts as a back pointer to the original bytecode." A90 9:26-29, A91 11:1-3. The system simply "replaces the go_native bytecode . . . with the original bytecode" that had been stored. A90-91 10:64-11:1.

The specification identifies alternative embodiments. For example, "the number of bytes or virtual machine instructions that are saved (or overwritten) may be varied in different embodiments and may depend on the virtual machine instructions themselves." A89 7:18-22; *see also* A90 9:36-40. Alternatively, the method may involve overwriting only a "portion" of an instruction, such as the first byte. A90 9:24-26; *see also* A89 7:7-22.

Independent claim 15 is representative of challenged claims 2-4, 16, and 18-21. It recites:

> 15. In a computer system, a method for increasing the execution speed of virtual machine instructions at runtime, the method comprising:
>
> receiving a first virtual machine instruction;
>
> generating, at runtime, a new virtual machine instruction that represents or references one or more native machine instructions that can be executed instead of the first virtual machine instruction;
>
> overwriting, at runtime, the first virtual machine instruction with the new virtual machine instruction; and
>
> executing the new virtual machine instruction and the one or more native machine instructions instead of the first virtual machine instruction.

A95. Independent claims 1 and 8 lack the "overwriting" step but include the steps of generating a new virtual machine instruction that references native instructions and executing the new instruction instead of the preexisting instruction. A92.

B. Magnusson And "Partial Translation"

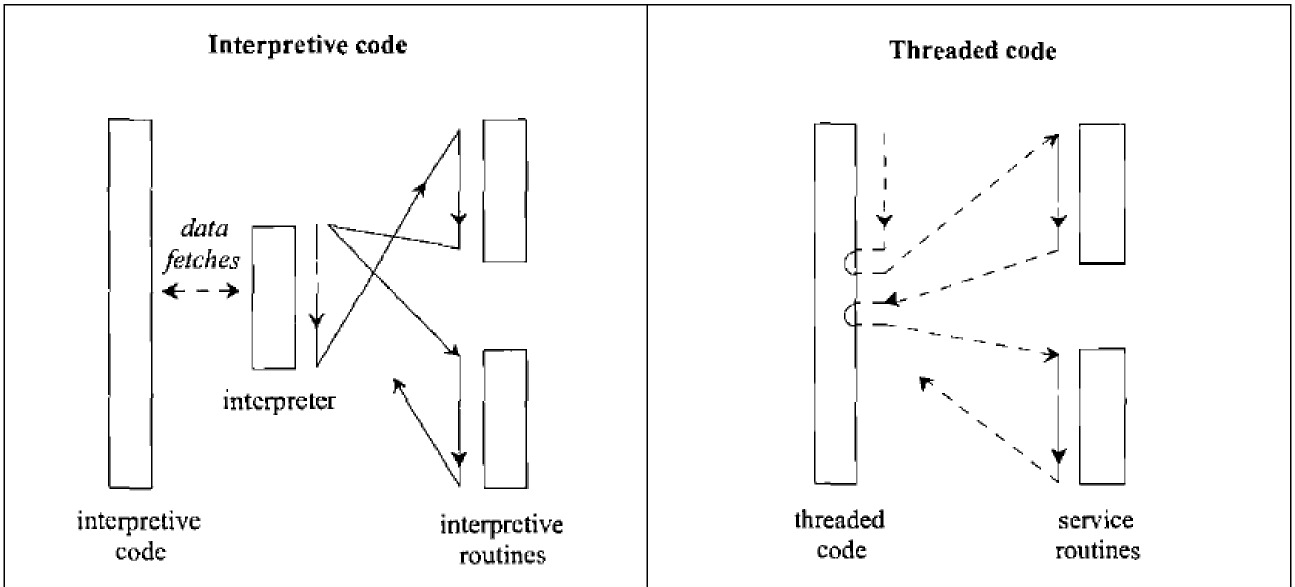
Magnusson is a technical report published nearly four years before the '205 patent application was filed. *See* A910, A912. It describes the use of interpreters to simulate the performance of computer architectures and system software, explaining that "simulation allows a program written for a particular machine to be executed on a dissimilar machine." A912. As with the '205 patent, Magnusson acknowledged the "slowdown" caused by "interpretive translation" and proposed

"a hybrid approach" as a solution. A912. The hybrid approach—called "partial translation"—"combine[s] the performance benefits of direct execution (running generated native code) with the flexibility and accuracy of interpretation." A917.

1. Magnusson's Discussion Of The Prior Art

Magnusson explained that simulation involves testing a program designed for one computer (the target machine) by executing it "in a simulation environment" on another computer (the host machine). A912. Some early "simulators actually translated the target code to host code"—that is, converted the instructions that are simulated into instructions capable of being executed by the machine running the simulation. *E.g.*, A914. Magnusson discusses these well-known simulation techniques in detail. *See* A914-15.

In the incremental translation approach, the simulator "[t]ranslat[ed] individual instructions to native code." A914. By contrast, the block translation approach involved "translat[ing] a program in parts or in its entirety" to native code for the host machine. A914. The starting point for block translation could be "source code, assembly code, or object code." A915. A third approach was interpretive translation: "target code is translated into a format that is easier to interpret, and this format is then interpreted." A915. Similar to the '205 patent, Magnusson recognized the performance drawbacks of interpreters. *See* A915; A918.



A915 (excerpt of Figure 3)

Magnusson explained that "[i]nterpretive translation gets a performance boost by threading the interpreted code," a technique "first described" in 1973. A915. Magnusson's Figure 3 again illustrates this concept. In a "threaded code" interpreter, "the task of the interpreter is done by the service routines. Instead of returning to a central interpreter, the epilogue of each service routine fetches sufficient information from the threaded code to be able to immediately jump to the corresponding service routine." A915; A895-96 ¶¶ 23-25. In other words, as Google's expert explained, "a threaded code model operates by replacing a virtual instruction with a reference to a service routine that performs the semantics of the instruction." A895 ¶ 25.

2. Magnusson's Hybrid Interpreter

From these existing methods, Magnusson concluded that "an interpretive approach has a high level of functionality" but "direct execution is the fastest approach." A925. Magnusson proposed a solution called "partial translation as a hybrid of the two." A925. Partial translation employs "the best of both approaches in a single design," A916, and "will perform considerably better" than either block translation or interpretive translation alone, A912.

Magnusson illustrates the solution in Figure 4:

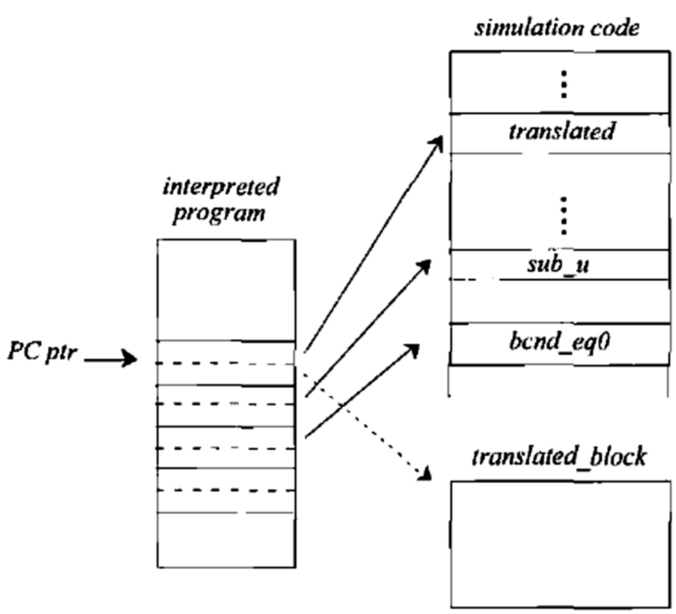


Figure 4 - Partial translation

A917. The box labeled "interpreted program" is the threaded code. *See* A918. Each 64-bit instruction in the interpreted program is split by a dashed line indicating two 32-bit sections, the first of which "points to the code that simulates the corresponding instruction" while the second "contain[s] parameters for the

instruction." A917; A893 ¶ 18. The box labeled "simulation code" represents service routines, such as "translated" and "sub_u," that simulate corresponding instructions in the interpreted program. A917. The corresponding instructions are identified in the text as "TRANSLATED" and "subu." *See* A918, A922. The "translated_block" box contains native code that can be executed instead of the virtual machine instructions in the interpreted program. *See* A917.

Magnusson identifies "two ways of combining the threaded code model with block translation": (1) include "a direct pointer to the compiled block" of native code or (2) "introduce a new instruction" called "TRANSLATED" in the stream of the interpreted program. A918. "This instruction takes as a parameter a pointer to the translated block, and handles generic entry/exit issues." A918. When the interpreter encounters the TRANSLATED instruction, it jumps to a "run-time generated block of" native machine instructions and executes them. A917; s*ee also* A894 ¶ 22.

Google's expert explained that Figure 4 shows the TRANSLATED instruction after it is "written into the interpreted program." A894 ¶ 19. The bits "located at the 'PC ptr' location point[] to the code that simulates the 'translated' instruction" while the "second 32-bits of the instruction located at the 'PC ptr' location" points "to the 'translated_block.'" A893-94 ¶ 19.

Magnusson provides pseudo-code examples that further demonstrate how the program re-enters the interpreted code stream following execution of the native code in the translated block. *See* A919-20. In one of these "exit" scenarios, the system stores the original intermediate instruction replaced by the translate instruction so that it can re-load the instruction and "branch[] to the interpretation code." *See* A920 (exit_0 scenario); A896-97. In another scenario, labeled "exit_d," the Magnusson system encounters invalid code, regenerates the original intermediate instruction, and re-enters the interpreted code. *See* A920; A897.

Magnusson further discloses actual code from a "prototype implementation." A922-23. Magnusson "effectively implemented" the prototype and produced "a substantial performance improvement over a purely interpretive approach, with no loss of functionality." A925. Although the prototype simulates object code for a M88100 machine on a SPARC machine, the partial translation approach "is processor independent." A918; A917; A896 ¶ 26.

C. The Reexamination Proceeding

The PTO found a substantial new question of patentability regarding original claims 1-4 and 8 of the '205 patent and new claims 15-16 and 18-21 (which Oracle added during the reexamination). *See* A774-75; A971. The Examiner rejected all challenged claims as anticipated by Magnusson. *See* A20-47. As relevant here, Oracle had opposed the rejections on two grounds: first, that Magnusson fails to

show the "overwriting" step of several challenged claims; and second, that Magnusson is not an enabling reference as to any claims. *See* A1008-23; *see also* A1038-55 (Google's response).

In a thorough right of appeal notice, the Examiner disagreed with Oracle. With respect to "overwriting," the Examiner concluded that "'it is clear that the first virtual machine instruction has been replaced by the TRANSLATED instruction.'" A44 (quoting A1050). The Examiner credited the declaration of Google's expert and adopted Google's reasoning, finding that "'Figure 4 of the Magnusson reference illustrates the introduction of a new instruction into the intermediate code; this introduction must overwrite existing code, else there would be no way to jump to the translated block.'" A44 (quoting A1050). The Examiner further agreed that Magnusson's "'various exit scenarios discuss storing the "old" first virtual machine instruction,'" and later restoring it, precisely because the instruction had been overwritten. A44 (quoting A1050).

After considering "the entire record," the Examiner also agreed with Google that Magnusson is an enabling reference. A23. Among other findings, the Examiner noted that "Magnusson did in fact provide a working prototype," A34, and Oracle "ignore[d] the fact that Magnusson described how the TRANSLATED instruction works in terms of how it is only slightly different from the well-known threaded code model," A30. The Examiner therefore rejected Oracle's apparent

position that a person of ordinary skill "would have required a disclosure of error-free code examples, explicit details on how to add a new instruction to existing code, and drawings that provide the exact same detail as the accompanying text in order to avoid undue experimentation . . . ." A41.

Oracle raised the same arguments before the Board. *See* A1166-89. In its rebuttal brief, Oracle also disputed, for the first time, the meaning of the term "overwriting." A1277-78. Oracle argued that Magnusson did not anticipate its claims even if Google "was correct that the TRANSLATED instruction overwrites preexisting code" because the claims "require[] that a new virtual machine instruction overwrite a specific instruction." A1278 (emphasis in original). Oracle argued that Magnusson "shows at best . . . merely preexisting code being replaced." A1278 (emphasis in original). Because Oracle presented this new claim construction position for the first time in its Rebuttal Brief, neither the Examiner nor Google could respond.

The Board "sustain[ed] the Examiner's rejections of claims 1-4, 8, 15, 16, and 18-21" in a written decision. A12. Applying the "broadest reasonable meaning" standard, the Board rejected Oracle's argument that "overwriting" means a specific kind of overwriting. *See* A7-8. The Board acknowledged a "'heavy presumption'" that a term takes its ordinary meaning. A7 (quoting *CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1366 (Fed. Cir. 2002)). Noting that

Oracle had pointed to no "clear definition" in the specification, the Board reviewed the specification and held that "overwriting" means "the act of replacing some information in a computer file with new information, rather than literally writing over an existing information." A8. The Board found that Magnusson satisfies the "overwriting" step because the introduction of the TRANSLATED instruction "replaces instructions within the intermediate code with new information." A9.

The Board also dismissed Oracle's enablement challenge, endorsing the Examiner's extensive discussion of the issue. *See* A11. The Board "considered the record" and found "that the arguments and evidence submitted by [Oracle] are not sufficient to overcome the presumption of enablement relied upon by the Examiner." A11.

SUMMARY OF ARGUMENT

The Board correctly upheld the Examiner's determination that Magnusson anticipated all challenged claims of the '205 patent.

I.A. The Board correctly construed "overwriting" to mean "the act of replacing some information in a computer file with new information, rather than literally writing over an existing information." A8. This construction reflects the term's ordinary meaning, which is "to replace information in (a computer file) with new information" (Merriam-Webster Dictionary (2014), *at* http://www.merriam-

webster.com/dictionary/overwrite), a meaning that is broad enough to encompass but not be limited to literally writing over.

It also reflects the patent's specification, which uses "overwrite" and "replace" interchangeably. For example, the written description states that the "new virtual machine instruction" (*i.e.*, the go_native bytecode that references native code for the computer to execute) "*replaces or overwrites* the initial virtual machine instruction of the selected portion of the function." A89 7:26-28 (emphasis added). At another point, when referring to the mechanism for restoring the initial virtual machine instruction, the written description uses only the word "replaces" to refer to acts that it elsewhere describes as "overwriting." *See* A90-91 10:64-11:1. Such interchangeable usage is strong evidence that the two words mean the same thing. *See Solvay S.A. v. Honeywell Int'l, Inc.*, 622 F.3d 1367, 1380-82 (Fed. Cir. 2010).

Oracle re-writes the Board's construction in an attempt to refute it. Contrary to Oracle's repeated assertions, the Board did not construe "overwriting" to mean "*not* writing over." Oracle Br. 23 (emphasis in original). The Board held that "overwriting" is *broader* than "literally writing over." A8. In contrast, Oracle simplistically splits "overwriting" into two words, reorders them as "writing over," and asserts that is the plain meaning and, moreover, the broadest reasonable interpretation. Oracle provides no support, only attorney argument, for this

supposed "plain meaning." The word is not self-defining. Ordinary principles of claim construction require consideration of *all* of the evidence, not isolated words in the abstract. Even on its own terms, Oracle's attorney argument suggests at most that "overwriting" may have two plain meanings—a broad ordinary definition (the Board's) that encompasses a more narrow meaning (Oracle's). The Board correctly chose the broadest reasonable construction.

B. Because the Board's construction of "overwriting" is correct, this Court should affirm. Oracle does not challenge the Board's factual finding that Magnusson discloses "overwriting" under its construction. Even if the Court were to reverse the Board's claim construction, it should affirm or, at most, remand to allow the Board to reconsider the issue. No remand is necessary, however, in view of the one-sided nature of the evidence.

As the Examiner determined, "'Figure 4 of the Magnusson reference illustrates the introduction of a new instruction into the intermediate code; this introduction must overwrite existing code, else there would be no way to jump to the translated block.'" A44 (quoting A1050). This finding is supported by Magnusson's use of the "threaded code model" as well as code examples that describe putting off translation "until run-time." As Google's expert explained, those aspects of Magnusson's partial translation separately and together require that "an original instruction must be overwritten." A896 ¶ 26.

Magnusson's "exit" scenario code confirms the point. These scenarios demonstrate that the partial translation system stores the original virtual machine instruction for the purpose of restoring it at a later time. In construing "overwrite," Oracle itself argues that the fact that "the original virtual machine instruction must be stored elsewhere" in the '205 patent shows that it has been "literally written over by a new virtual machine instruction." Oracle Br. 12. Oracle cannot argue the opposite with respect to Magnusson.

II. The Board also correctly held that Magnusson is an enabling reference. Magnusson's disclosure goes well beyond what would be required to "teach one of ordinary skill in the art to make or carry out the claimed invention without undue experimentation." *Minn. Mining & Mfg. Co. v. Chemque, Inc.*, 303 F.3d 1294, 1306 (Fed. Cir. 2002). It discusses the state of the art, provides extensive descriptions and code, and discloses a working prototype.

Oracle's challenges amount to flyspecking. Oracle criticizes Magnusson for not providing code showing the introduction of the TRANSLATE instruction. *See* Oracle Br. 53. No code is necessary because, as the Examiner found, "an experienced programmer could, without unreasonable effort, write a program to carry out the invention." *Northern Telecom, Inc. v. Datapoint Corp.*, 908 F.2d 931, 941 (Fed. Cir. 1990) (per curiam).

Oracle also points to a supposed typographical error in the code examples that Magnusson does provide. *See* Oracle Br. 55-56. As Oracle admits, however, any discrepancy is resolved by the comment notation next to the code. Oracle does not even challenge the Examiner's finding that "'[o]ne of ordinary skill in the art would have clearly understood . . . the point of this instruction.'" A34 (quoting A1047).

STANDARD OF REVIEW

This Court reviews the Board's factual findings for substantial evidence and legal conclusions de novo. *See In re Gartside*, 203 F.3d 1305, 1316 (Fed. Cir. 2000); *In re Elsner*, 381 F.3d 1125, 1127 (Fed. Cir. 2004). Claim construction is a question of law reviewed de novo. *See Lighting Ballast Control LLC v. Philips Elecs. N. Am. Corp.*, 744 F.3d 1272, 1276-77 (Fed. Cir. 2014) (en banc). Anticipation is a question of fact reviewed for substantial evidence. *See In re Gleave*, 560 F.3d 1331, 1334-35 (Fed. Cir. 2009). Enablement is a question of law based on underlying factual findings. *See Minn. Mining & Mfg.*, 303 F.3d at 1301. "A finding is supported by substantial evidence if a reasonable mind might accept it as adequate to support the finding." *In re Adler*, 723 F.3d 1322, 1325 (Fed. Cir. 2013).

ARGUMENT

I. MAGNUSSON ANTICIPATED THE ’205 PATENT.

Oracle now concedes that Magnusson discloses each and every element of claims 1 and 8 of the ’205 patent. And for good reason: Magnusson and the ’205 patent share undeniable similarities. Both seek to improve the speed of interpreters by using a “hybrid” of virtual and native machine instructions that combines the speed of native code with the flexibility of virtual machines. *E.g.*, A72; A912; *see also* A91 11:16-19; A917. Accordingly, there is no dispute that Magnusson discloses the steps of generating and executing a new virtual machine instruction that references native machine instructions for execution instead of an original virtual machine instruction. *See* A94 (claim 1; claim 8).

As to the remaining claims, Oracle contends that Magnusson fails to disclose a single claim element—the “overwriting” step. A7. Oracle is wrong. The Board correctly construed “overwriting” according to the term’s ordinary meaning, the patent’s specification, and the “broadest reasonable construction” standard. In any event, even under Oracle’s crabbed construction of the term, Magnusson discloses the “overwriting” step, as the Examiner found.

A. The Board Correctly Construed “Overwriting.”

Oracle paints the claim construction issue as a simple matter of breaking the word “overwriting” into its constituent parts and reordering them as “writing

over." *See, e.g.*, Oracle Br. 17. That is wrong. Overwriting is not a self-defining term, and Oracle's simplistic approach would divorce the term from its plain meaning and its usage in the specification. The Board correctly construed the term in harmony with both.

1. The Plain Meaning Of "Overwriting" Includes Replacing Information.

The Board's construction of "overwriting" reflects the term's ordinary and customary meaning. *See Phillips v. AWH Corp.*, 415 F.3d 1303, 1312-13 (Fed. Cir. 2005) (en banc). The Board construed the term to mean "replacing some information in a computer file with new information, rather than literally writing over an existing information." A8. That is exactly its ordinary meaning: "to replace information in (a computer file) with new information." Merriam-Webster, *at* http://www.merriam-webster.com/dictionary/overwrite. Put another way, the term means "[t]o record (data) in a file, on a disk, etc. in such a way as to replace data that is already there." Webster's New World College Dictionary (4th ed. 2002).

The specification supports this ordinary understanding by using "overwrite" and "replace" interchangeably. For example, the written description describes the need for "a new virtual machine instruction" that references native code to be executed instead one or more virtual machine instructions. A89 7:23-24. According to the patent, "[t]his new virtual machine instruction *replaces or*

overwrites the initial virtual machine instruction of the selected portion of the function." A89 7:26-28 (emphasis added). The phrase "replaces or overwrites" indicates the words are synonyms. *See 01 Communique Lab., Inc. v. LogMeIn, Inc.*, 687 F.3d 1292, 1296 (Fed. Cir. 2012) (holding that phrase "data communication program or facility" used terms "facility" and "program" interchangeably).

The specification confirms the point by using the term "replaces" in one passage to refer to acts elsewhere described as "overwriting." It states that, to reverse the process of overwriting the original virtual machine instruction with the go_native instruction, "the system *replaces* the go_native" instruction "with the original" instruction and thereby restores it to its original memory address. A90 10:64-67 (emphasis added); *see also* A81. The specification's interchangeable use of terms is strong evidence that "overwriting" includes "replacing." *See Solvay*, 622 F.3d at 1380-82 (holding that claim was "not limited to 'isolating' *only*" when "[t]he patent specification uses the terms 'isolating,' 'separating,' and 'drawing off' interchangeably"). The fact that this interchangeable usage aligns with the term's ordinary meaning removes any doubt. *See Thorner v. Sony Computer Entm't Am. LLC*, 669 F.3d 1362, 1365 (Fed. Cir. 2012) (ordinary meaning controls in absence of clear contrary usage in patent).

2. Oracle Mischaracterizes The Board's Construction.

Seeking to undo the Board's construction, Oracle re-writes it, stating that the Board construed "overwriting" to mean "*not* writing over." Oracle Br. 23 (emphasis in original); *see also id.* at 2, 17, 24. That is a gross mischaracterization. The Board did not hold that overwriting *excludes* "writing over," as Oracle contends. Rather, it construed the term to mean not just "literally writing over an existing information" but also the "act of replacing some information in a computer file with new information," A8. In other words, the Board held that "overwriting" is broader than "literally writing over." No fair reading of the Board's decision could support any other conclusion.

Oracle also accuses the Board of construing "overwriting" without warning. *See* Oracle Br. 3, 16, 25. In fact, Oracle prompted the Board to construe the term by arguing, for the first time in its rebuttal brief, that "overwriting" required more than a showing that an "instruction overwrites preexisting code" or that "preexisting code [is] being replaced." A1278 (emphasis omitted). Having belatedly raised this claim-construction issue, at a time when it was too late for Google or the Examiner to respond, Oracle is the last party that should be heard to complain about the Board's willingness to consider the question.

Oracle's criticisms of the Board's reasoning are just as inaccurate and unpersuasive. Oracle argues that the Board "never paused to consider the ordinary

and customary meaning" of the term. Oracle Br. 24. Yet Oracle concedes that the Board acknowledged "a 'heavy presumption' that a claim term carries its ordinary and customary meaning." A7 (quoting *CCS Fitness*, 288 F.3d at 1366). Although the Board did not cite the dictionary definitions discussed above, the fact that it adopted them nearly verbatim is hardly a coincidence; at a bare minimum, it confirms that the Board adopted a plain-meaning interpretation. This Court is free to take judicial notice of those definitions. *See Pyles v. Merit Sys. Prot. Bd.*, 45 F.3d 411, 415 (Fed. Cir. 1995).

In contrast, Oracle conjures its alternative construction out of thin air. Without citation, it asserts that "[o]ne of ordinary skill in the art would not have understood 'overwriting' to mean anything other than writing over." Oracle Br. 24. Attorney argument, and starting sentences with references to persons skilled in the art, are no substitute for actual evidence. The closest Oracle comes to identifying evidence of its purported plain meaning is its argument that "the claim term 'overwriting' could not be clearer"—that is, that the word is self-defining. *Id.* at 23. It is not. Many words are more than, or different from, the sum of their parts. Override, underwrite, underdog, and brainstorm are a few other examples. And the dictionaries cited above make clear that the term "overwriting" is not defined simply by breaking it apart and reordering the parts.

Even if Oracle could establish a plausible ordinary meaning purely through attorney argument (which it cannot), that would mean only that the term has two potential plain meanings. In that circumstance, the "broadest reasonable construction" would prevail. *Phillips*, 415 F.3d at 1316. This standard "'serves the public interest by reducing the possibility that claims, finally allowed, will be given broader scope than is justified.'" *In re Am. Acad. of Sci. Tech Ctr.*, 367 F.3d 1359, 1364 (Fed. Cir. 2004) (quoting *In re Yamamoto*, 740 F.2d 1569, 1571 (Fed. Cir. 1984)). In other words, the law aims to prevent patentees from declining to define terms in the specification and then selectively narrowing them later—exactly what Oracle is attempting to do here.

The specification provides further confirmation that the Board construed the term correctly. Oracle looks to Figure 5 to show "that original virtual machine instruction 'BYTECODE 2' has been *written over* by a new virtual machine instruction 'GO_NATIVE #N.'" Oracle Br. 25 (emphasis in original). But no one disputes that literally writing over constitutes overwriting; the point, which Oracle does not dispute, is that neither the figure nor any accompanying text rises to the level of lexicography that defines overwriting to mean *only* literally writing over. It would therefore be improper to confine the claims to a preferred embodiment. *See, e.g.*, *Laryngeal Mask Co. v. Ambu*, 618 F.3d 1367, 1372 (Fed. Cir. 2010); *Resonate, Inc. v. Alteon Websystems, Inc.*, 338 F.3d 1360, 1367 (Fed. Cir. 2003).

Put another way, whether or not some language, "when viewed in isolation," might support Oracle's narrow construction is irrelevant because "the specification as a whole suggests a construction that is not so narrow" (*Am. Acad. of Sci. Tech Ctr.*, 367 F.3d at 1367)—especially under the broadest reasonable interpretation standard.

For that reason, although the Board agreed that Figure 5's "explicit illustration" depicted only bytecode 2 being overwritten, it correctly determined that the specification spoke more broadly because "bytecodes 3-5 are also being replaced by one or more native machine instructions." A8. Oracle repeatedly contends that bytecodes 3-5 are not overwritten, *see, e.g.*, Oracle Br. 27, 29, but the specification states "that the number of bytes or virtual machine instructions that are saved (or overwritten) may be varied" depending "on the virtual machine instructions themselves." A89 7:19-21. And as discussed above, the written description uses "overwrite" and "replace" interchangeably. *See* pp. 23-24, *supra*.

Oracle also contends that the Board's reasoning violates claim differentiation principles by equating overwriting (as used in claim 2) with executing a new virtual machine instruction instead of a first virtual machine instruction (as required by claim 1). Oracle Br. 30-31. This argument misses the mark. Far from equating the two claims, the Board merely rejected Oracle's belated claim-construction argument in its Rebuttal Brief that the "overwriting"

step requires not only that an "instruction overwrites preexisting code" but also "requires that a new virtual machine instruction overwrite a specific instruction." A1278. Under that theory, Figure 2 would embody the overwriting step only if the go_native instruction literally overwrote bytecodes 3-5 along with bytecode 2. *See* A1278 ("Magnusson's FIG. 4 would have had to show that the TRANSLATED instruction overwrote the specific code whose function the TRANSLATED instruction was to execute instead."). The Board correctly concluded that "overwriting" means all overwriting and not the narrower subset proposed by Oracle.

That construction does not conflate claims 1 and 2 because it concerns how information is *recorded*, not how the instructions contained in that information are *executed*. There is nothing inconsistent about claim 2's express recitation of an action that may be implied by the actions required in claim 1. Moreover, claim 2 clarifies that the new virtual machine instruction will "specify[] execution" of the native machine instruction in addition to representing or referencing it. These "different words" also suggest "different meanings and scope." *Karlin Tech., Inc. v. Surgical Dynamics, Inc.*, 177 F.3d 968, 971-72 (Fed. Cir. 1999). Claim 2 is simply not co-extensive with claim 1 under the Boards construction, as Oracle contends.

3. Substantial Evidence Supports A Finding Of Anticipation Under the Board's Construction of "Overwriting."

Because the Board correctly construed "overwriting," this Court should affirm. The Board found "that Magnusson discloses the use of intermediate code instructions for an emulated processor and then jumping to a native code (SPARC) block upon encountering a TRANSLATED instruction." A9. As a result, Magnusson "replaces instructions within the intermediate code with new information," satisfying the "overwriting" step. A9. Oracle does not challenge that factual conclusion, which is fully supported by Magnusson itself and the testimony of Google's expert. *See* A840-41 ¶ 15; A896 ¶ 26.

Indeed, all of Oracle's arguments regarding Magnusson's disclosure rest on its misguided construction of "overwriting." Thus, if the Court affirms the Board's construction, there is no reason to reach those arguments. *See* Oracle Br. 32-51.

B. Substantial Evidence Supports A Finding Of Anticipation Under Oracle's Construction of "Overwriting."

Even if the Court were to adopt Oracle's narrow construction, it should affirm or, at most, remand to allow the Board to weigh the evidence in view of the new construction. No remand is necessary, however, in view of the existing evidence. Oracle admits that the Examiner determined that Magnusson discloses literally writing over existing information. Oracle Br. 24. That finding is not altered by the precise claim construction, which is presumably why Oracle did not

even raise a claim-construction issue until its rebuttal brief before the Board. *See* A1278.

1. The TRANSLATED Instruction Overwrites An Original Virtual Machine Instruction.

The Examiner determined that "'Figure 4 of the Magnusson reference illustrates the introduction of a new instruction into the intermediate code; this introduction must overwrite existing code, else there would be no way to jump to the translated block.'" A44 (quoting A1050). The Examiner found support for this conclusion in the text of Magnusson and the declaration of Google's expert.

As an initial matter, Figure 4 (at right) shows the TRANSLATED instruction in the interpreted program. A917. As Google's expert explained, the instruction located at "PC ptr" in Figure 4 is a 64-bit TRANSLATED instruction "after it has been . . . written into the interpreted program." A893-94 ¶ 19.

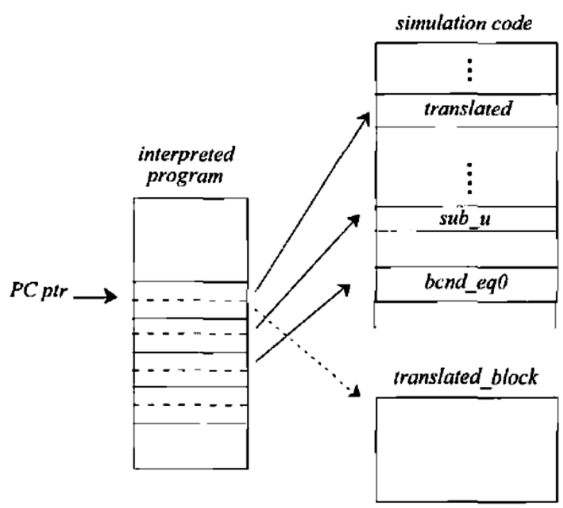


Figure 4 - Partial translation

The description accompanying this figure explains that the TRANSLATED instruction overwrites the existing instruction. Partial translation "combin[es] the threaded code model with block translation" by "introduc[ing] a new instruction, TRANSLATED," into the interpreted program stream. A918. The reference to the

"threaded code model" is important because "in order for a 'threaded code model' to operate, an original instruction must be overwritten," as Google's expert explained. A896 ¶ 26. One of skill in the art would read this passage with an understanding of "the operating principles" of that model, which Magnusson itself describes in detail. A896 ¶ 26.

There is more. Magnusson provided a separate code example for its prototype, for which "translation must be put off until run-time." A915, A917. As Google's expert explained, that means that the introduction of the TRANSLATED instruction "must have also occurred at runtime" because "[t]he translated/compiled block could not exist until runtime." A896 ¶ 26. One of skill would have understood from this description that the TRANSLATED instruction in the interpreted program "is the result of the simulator's overwriting—at runtime—an original instruction with the newly-generated TRANSLATED instruction." A896 ¶ 26. This too is the result of the "threaded code model." A896 ¶ 26.

Magnusson's pseudo-code "exit" scenarios further support the Examiner's findings. For example, Google's expert explained that the exit_0 routine "clarifies that the original virtual machine instruction is stored in [a variable called] <CONST>" and that "[t]he reason that Magnusson stores the original instruction in the <CONST> constant is because . . . the TRANSLATED instruction overwrites

the original virtual machine instruction within the threaded code instruction sequence (i.e., the 'interpreted program' of Figure 4)." A897 ¶ 30. This exit scenario loads the "'value' with the original instruction that was stored in the <CONST> constant" and then branches "to the interpretation code.'" A897 ¶ 30 (quoting A920). In other words, to return to the interpreted program, the system must first restore the original virtual machine instruction because it no longer exists after being overwritten.

Oracle contends that "Google has imagined that an 'original virtual machine instruction' is stored in the <CONST> value." Oracle Br. 43. But Oracle agreed below that "<CONST> contains information about the original instruction." A1020. Oracle also contends that storing such information "is not *necessarily* due to overwriting." Oracle Br. 43 (emphasis in original). In its claim construction argument, however, Oracle asserts that the fact that "the original virtual machine instruction must be stored elsewhere" (in an embodiment of the '205 patent) shows that it has been "literally written over by a new virtual machine instruction." Oracle Br. 12; *see also* A89 8:38-53; A90-91 10:55-11:1. Oracle is, in other words, seeking one claim construction for purposes of the '205 patent but then implicitly relying on a different one in distinguishing Magnusson. Oracle cannot have it both ways. What is true for the '205 patent is equally true for Magnusson. *Cf. Amazon.com, Inc. v. Barnesandnoble.com, Inc.*, 239 F.3d 1343, 1351 (Fed. Cir.

2001) (patent's meaning may not be twisted "like a nose of wax" for different purposes (quotation marks and citations omitted)).

Oracle's assertion that the exit code does not itself *perform* the overwriting is a straw man. *See* Oracle Br. 41-46. Neither Google nor its expert ever argued that it did. The point is that Magnusson's exit scenarios provide further confirmation that the TRANSLATED instruction overwrites an original instruction in the intermediate format instruction stream. A865-68; A896-98 ¶¶ 27-35; A1050.

2. Oracle's Counter-Arguments Are Legally And Factually Wrong.

Oracle does not meaningfully address this evidence. Instead, it contends that Magnusson could not anticipate because it "does not even mention 'overwriting.'" Oracle Br. 34-35. Anticipation does not turn on magic words, however, and Magnusson "need not satisfy an *ipsissimis verbis* [identical words] test." *Gleave*, 560 F.3d at 1334. Rather, a reference anticipates a claim "if a person of ordinary skill in the art would understand the [reference] as disclosing [the] elements." *Helifix Ltd. v. Blok-Lok, Ltd.*, 208 F.3d 1339, 1347 (Fed. Cir. 2000); *see also Arthrocare Corp. v. Smith & Nephew, Inc.*, 406 F.3d 1365, 1373-74 (Fed. Cir. 2005).

Similarly, Oracle discounts Magnusson's Figure 4 because it does not depict "the term TRANSLATED." Oracle Br. 38. Figure 4 cannot be divorced from its

surrounding text, however, which describes the introduction of the TRANSLATED instruction in more than adequate detail. Oracle accuses Google and the Examiner of "stitch[ing] together various isolated portions of Magnusson." Oracle Br. 35-41. But "[a] reference must be considered for everything that it teaches." *In re Applied Materials, Inc.*, 692 F.3d 1289, 1298 (Fed. Cir. 2012). Not only was the Examiner free to rely on all aspects of Magnusson (a 17-page technical report), it would have been error to ignore them.

In any event, Oracle's argument regarding Figure 4 is purely one of form over substance because Oracle and its expert both conceded below that the TRANSLATED instruction was "[i]n the intermediate format instruction stream." A1020; *see also* A1032 ("Magnusson's system has a choice to either introduce a normal intermediate format instruction or a TRANSLATED instruction *into the intermediate instruction stream*." (emphasis added)). Oracle's opening brief says so as well. *See* Oracle Br. 40. The introduction of the TRANSLATED instruction into that stream overwrites the preexisting instruction, for all the reasons discussed above.

Oracle responds that "to 'introduce' is not the same as 'overwriting'" and "[a]ll that Magnusson is stating is that a new function or operation can be put into use—and nothing more." Oracle Br. 34. Oracle provides no evidentiary support for that argument. Oracle's attorney argument cannot stand against Magnusson's

text, its code examples, and the testimony of Google's expert. As the Examiner observed, Oracle "ignores the fact that Magnusson described how the TRANSLATED instruction works in terms of how it is only slightly different from the well-known threaded code model." A30. Ample evidence supports the Examiner's factual findings. *See* A893 ¶ 17.

Oracle next objects that "Google improperly used expert testimony to support its anticipation position . . . ." Oracle Br. 47. No principle of law supports such an extreme position. As this Court has sensibly held, "expert testimony may be critical, for example, to establish the existence of certain features in the prior art." *Wyers v. Master Lock Co.*, 616 F.3d 1231, 1240 n.5 (Fed. Cir. 2010). That is especially true for the fact-bound anticipation inquiry, which depends on viewing a reference from the point of view of one of skill in the art. *See Arthrocare*, 406 F.3d at 1373-74. Indeed, Google presented an expert declaration to rebut the mischaracterizations in the declaration of *Oracle's* expert. A837-44. Having initiated the battle of experts, Oracle cannot cry foul now that the Examiner sided with Google's expert.

Nor did Google use expert testimony "to supply" any missing limitations or to demonstrate inherent anticipation. This argument is simply a reprise of Oracle's argument that Magnusson does not use the word "overwriting." Oracle Br. 39-41, 49-51. For the same reason that a reference need not disclose magic words, a party

may offer expert testimony to show "that a person of skill in the art would understand" a reference to disclose a limitation, even where the reference does not "explicitly identify" the limitation. *Arthrocare*, 406 F.3d at 1373-74. The Examiner was free to credit the testimony of Google's expert for that purpose.

Regardless, Magnusson would anticipate even under an inherent anticipation theory. Google's expert did not "merely assert[] possibilities for how Magnusson's TRANSLATED instruction is introduced," as Oracle contends. Oracle Br. 40. Rather, he stated that "an original instruction *must* be overwritten by . . . the TRANSLATED instruction" and "[t]his is the *only way* that Magnusson's threaded code model would operate." A896 ¶ 26 (emphasis added).

II. MAGNUSSON ENABLES.

"[A] prior art printed publication . . . is presumptively enabling barring any showing to the contrary." *In re Antor Media Corp.*, 689 F.3d 1282, 1288 (Fed. Cir. 2012). In twenty pages of detailed findings, the Examiner "found that *Magnusson* is enabled, with or without any initial presumption of enablement." A22; *see also* A21-42. The Board likewise "considered the record" and affirmed. A11. Oracle did not come close to carrying its burden before the Board, and it is no closer to overcoming the substantial evidence standard of review on appeal.

A. The Board Properly Reviewed The Entire Record And Resolved Factual Disputes Against Oracle.

The evidence of enablement is overwhelming. A reference is enabling if it "teach[es] one of ordinary skill in the art to make or carry out the claimed invention without undue experimentation"—a legal standard Oracle does not dispute (Oracle Br. 57). *Minn. Mining & Mfg.*, 303 F.3d at 1301. Magnusson goes well beyond that minimal requirement. Magnusson details the state of the art at the time it was published, A914-16; it describes a new solution with text, figures, exemplary code, and a description of a working prototype, A917-24; and it supports all of that disclosure with evidence of "benchmark" testing, A922. As Google's expert explained, the reference sets forth "a significant achievement and a significant proof of concept." A898 ¶ 37. The Board and the Examiner were entitled to rely on Magnusson's disclosure and the testimony of Google's expert to conclude that Magnusson enables skilled artisans to carry out the claimed technology without undue experimentation.

Oracle's position depends on holding Magnusson to a higher standard than this Court's cases require. As the Examiner explained, Oracle implicitly contends that one of ordinary skill "would have required a disclosure of error-free code examples, explicit details on how to add a new instruction to existing code, and drawings that provide the exact same detail as the accompanying text in order to

avoid undue experimentation." A41. Contrary to Oracle's argument, the law neither insists on perfection nor disregards the knowledge of skilled artisans.

1. The Enablement Standard Does Not Require Code.

Oracle first contends that Magnusson is not enabling because "the examples lack any code showing the introduction of the TRANSLATED instruction." Oracle Br. 53. Oracle cites no precedent to support its argument, and this Court has never held that a prior art reference must disclose code to be enabling. Rather, "[w]hen the challenged subject matter is a computer program that implements a claimed device or method, enablement is determined from the viewpoint of a skilled programmer using the knowledge and skill with which such a person is charged." *Northern Telecom*, 908 F.2d at 941. So long as "an experienced programmer could, without unreasonable effort, write a program to carry out the invention," the enablement requirement is satisfied. *Id.*; *see also In re Hayes Microcomputer Prods., Inc. Patent Litig.*, 982 F.2d 1527, 1534 (Fed. Cir. 1992).

Magnusson is clearly enabling under that standard. *See* A31-32. As Google's expert explained, Magnusson discloses that the TRANSLATED instruction is generated as a 64-bit instruction. A893 ¶ 18. A person of ordinary skill reading this description would learn that the first 32 bits point to the code that simulates the instruction and the second 32 bits point to the "translated_block" of native machine instructions depicted in Magnusson's Figure 4. A893-94 ¶¶ 18-19.

Given "such detail," it is clear "that one of ordinary skill in the art . . . could easily implement Magnusson's system." A894 ¶ 20.

A programmer of ordinary skill would also have been familiar with relevant background knowledge in the art. As Google's expert explained, such a person would have found "no significant technical distinction between" the use of the term "introducing" in Magnusson and the use of the term "generating" in the '205 patent. A893 ¶ 17. Skilled artisans would have been "familiar with compilers" and understood "that 'generating' an instruction refers to the compiler's formation of the syntactic representation of an instruction and outputting that representation to a file or other data structure for subsequent use." A893 ¶ 17. Magnusson uses the term "introduce" to mean "the same thing" because the simulator "forms the syntactic representation of the TRANSLATE instruction" and then "outputs that instruction into the threaded code instruction sequence." A893 ¶ 17. Based on this knowledge alone, "it would be relatively straightforward for a skilled computer programmer" to introduce the TRANSLATE instruction using familiar skills. *Northern Telecom*, 908 F.2d at 941-42.

This evidence—which the Examiner adopted across the board—stands unrebutted. *See* A31-32. Oracle ignores it, arguing only that Magnusson lacks adequate code. Oracle Br. 52-56. But if code were necessary for Magnusson to be enabling, that would effectively mean that applicants would have to disclose

source code as a prerequisite for obtaining a patent on any computer-related invention (to satisfy 35 U.S.C. § 112's enablement requirement). Neither this Court nor the PTO requires that level of detail, which is presumably why the '205 patent does not include any code. In any event, Oracle never explains what additional information could have been provided by code beyond what was already known to skilled programmers. Indeed, Oracle never even mentions "the knowledge and skill" of one of skill in the art. *Northern Telecom*, 908 F.2d at 941. At most, Oracle is "merely pointing to some missing details." *In re Antor*, 689 F.3d at 1289. That alone is reason to affirm.

The only supposedly contrary evidence cited by Oracle is a conclusory reference to the testimony of its own expert. *See* Oracle Br. 54. The Board and the Examiner were entitled to reject that testimony. As this Court has held, "[t]he Board has broad discretion as to the weight to give to declarations offered in the course of prosecution." *Am. Acad. of Sci. Tech Ctr.*, 367 F.3d at 1368. In any event, that testimony merely reiterates Oracle's untenable position that a lack of code automatically entails a lack of enablement. *See* Oracle Br. 54.

2. Minor Errors Are Not Evidence Of Undue Experimentation.

Despite Oracle's hyperbolic assertion that Magnusson "is filled with discrepancies," *id.* at 13, its brief identifies only two supposed errors, *see id.* at 55-56. Although Oracle presented both of those factual arguments below, *compare*

A33-35, *with* Oracle Br. 55-56, they have not improved with age. Indeed, the only thing new is the highly deferential standard of review. "This court does not reweigh evidence on appeal, but rather determines whether substantial evidence supports the Board's fact findings." *In re NTP, Inc.*, 654 F.3d 1279, 1292 (Fed. Cir. 2011).

Oracle first points to an alleged discrepancy in the pseudo-code instruction "rPC = rPC + 24" and its associated comment "point to next instruction." A919. The Examiner found no confusion, adopting Google's reasoning in full. *See* A34. Citing the unambiguous, explanatory comment, the Examiner found that "'[o]ne of ordinary skill in the art would have clearly understood that the point of this instruction was to move the program counter to the next instruction after the translated host (*i.e.*, SPARC) code has run.'" A34 (quoting A1047); *see also* A1065 ¶¶ 18-21.

Oracle does not seem to dispute this. It argues only that the number "24" is unclear because it might be a typographical error. Oracle debates whether "Magnusson *meant* to point to the next instruction" or whether it "*does* move the pointer to the next instruction." Oracle Br. 55 (emphases added). That is a distinction without a difference for purposes of enablement. And Oracle does not even argue that a person of skill would have been incapable of implementing code designed to move a pointer to a next instruction (including doing so for the 64-bit

instruction set disclosed in Magnusson, A917). Resolving a typographical error in that circumstance would not even rise to the level of "some experimentation," much less undue experimentation. *Hybritech Inc. v. Monoclonal Antibodies, Inc.*, 802 F.2d 1367, 1384 (Fed. Cir. 1986). Thus, the Examiner's factual determination that a person of skill would have understood the code remains undisturbed. *See* A34.

Oracle's other alleged discrepancy relates to the same "rPC" value in other code. It therefore fails for the same reasons. Oracle contends that the "confusion" regarding rPC "carries over to the 'exit' block" in a separate piece of pseudo code. Oracle Br. 55. Because a person of skill in the art would not have been confused by "rPC + 24," no confusion could have carried over to other code segments.

Moreover, the Examiner pointed to the "actual SPARC assembly code implementation" to refute Oracle's assertion that the rPC value "'could not contain both the address of the service routine and the parameters required'" by the exit code. A34-34 (quoting A1011). According to the Examiner, this code "shows the first 32-bit portion of a next instruction being loaded into %g1, the second 32-bit portion . . . loaded into %g3, and dispatching the next instruction by the command 'jmp %g1.'" A34 (citing A922-24). In other words, the code shows "using only the first 32-bit portion of the loaded instruction." A34. The comment next to the pseudo-code confirms that the purpose of the "rOP = [rPC]" instruction is to "get

[the] next instruction specification." A920. Oracle now presents the same misunderstanding of the code while ignoring the Examiner's contrary factual findings. *See* Oracle Br. 55-56.

Accordingly, there is no support for Oracle's conclusion that these are "unresolvable discrepancies." Oracle Br. 56. And even if Oracle were correct that one of skill in the art would have to tweak the disclosed code to create a working prototype, that would not demonstrate lack of enablement. Even disclosures of inoperable devices may be enabling, and "proof of efficacy is not required for a prior art reference to be enabling for purposes of anticipation." *Impax Labs. Inc. v. Aventis Pharm. Inc.*, 468 F.3d 1366, 1383 (Fed. Cir. 2006).

B. Oracle's Other Arguments Are Wrong.

Oracle contends that "the Board largely followed the Examiner's and Google's improper comparison of the amount of disclosure in the '205 Patent with the amount of disclosure in Magnusson to determine Magnusson's enablement." Oracle Br. 52. To the contrary, the Board chastised *Oracle* for "engag[ing] in this comparison to attempt to demonstrate the superior nature of the disclosure within the '205 patent." A11. Indeed, Oracle makes the same comparison in its opening brief. *See* Oracle Br. 59-60. Oracle's reliance on this Court's decision in *Morsa* is therefore both ironic and irrelevant. *See In re Morsa*, 713 F.3d 104, 110-11 (Fed. Cir. 2013). The Board did not rely on a lack of disclosure in the patent-at-issue at

all, much less *in lieu of* addressing an applicant's "specific, concrete reasons why he believed the [prior art] at issue was not enabling." *Id.*

The only thing Oracle can point to is the Examiner's statement that the '205 patent's failure to include "the type of detail . . . that the patent owner now argues is necessary in Magnusson supports a finding that one skilled in the art would have known how to implement the features of Magnusson . . . ." A26 (citing *In re Epstein*, 32 F.3d 1559, 1568 (Fed. Cir. 1994)). That is not the kind of "head to head comparison" of Magnusson and the '205 patent that *Morsa* forbids. It is instead an analysis of *Oracle's arguments* regarding enablement. That is precisely what the Court approved in *Epstein* when it affirmed "the Board's observation that appellant did not provide the type of detail in his specification that he now argues is necessary in prior art references." 32 F.3d at 1568. The patentee's arguments "support[ed] the Board's finding," based principally on other evidence, "that one skilled in the art would have known how to implement the features of the references and would have concluded that the reference disclosures would have been enabling." *Id.*

Finally, Oracle argues that the Board "misappl[ied] the requisite level of skill." Oracle Br. 60-61. The argument is frivolous. Oracle identifies a single statement by the Examiner: that Oracle's "own argument appears to demonstrate that" Magnusson "would have been readily understandable despite" a supposed

inconsistency regarding Magnusson's description of run-time generated code. A28. This statement does not appear to have anything to do with any of Oracle's current arguments. It refers instead to a supposed discrepancy that Oracle asserted before the Board but has now abandoned. *See* A28 (citing A1017).

If anyone has misapplied the level of skill, it is Oracle. The Examiner agreed with Google's "assessment of the level of ordinary skill in the relevant art" and rejected the level of skill proposed by Oracle. *See* A40-41. In addition, the Examiner adopted the reasoning of Google's expert, who "clearly presented his opinion based on the level of ordinary skill in the art." A42 (citing A892-99 ¶¶ 10-12, 17, 20-23, 24, 26, 42; A1064-65 ¶¶ 13, 21). The Board affirmed these findings. A11. Those factual determinations, which Oracle does not contest, are amply supported by the record. *See* A40-41 (citing A892 ¶¶ 11-12, A86 2:27-31).

CONCLUSION

This Court should affirm the Board's conclusion that claims 1-4, 8, 15, 16, and 18-21 are anticipated by Magnusson.

DATED: July 28, 2014

Respectfully submitted,

/s/ Daryl L. Joseffer

Daryl L. Joseffer
Counsel of Record
KING & SPALDING LLP
1700 Pennsylvania Ave. NW
Washington, DC 20006

Scott T. Weingaertner
KING & SPALDING LLP
1185 Avenue of the Americas
New York, NY 10036

Robert T. Neufeld
KING & SPALDING LLP
1180 Peachtree St., NE
Atlanta, GA 30309

Adam M. Conrad
KING & SPALDING LLP
100 N. Tryon St., Ste. 3900
Charlotte, NC 28202

Brian C. Banner
KING & SPALDING LLP
401 Congress Ave., Ste. 3200
Austin, TX 78701

CERTIFICATE OF COMPLIANCE

Pursuant to Federal Rule of Appellate Procedure 32(a)(7)(C), the undersigned certifies that the foregoing brief, exclusive of the exempted portions as provided in Fed. R. App. P. 32(a)(7)(B)(iii) and Fed. Cir. R. 32(b), contains 9,756 words and therefore complies with the type-volume limitations of Fed. R. App. P. 28.1(e)(2)(A)(i).

DATED: July 28, 2014

/s/ Daryl L. Joseffer
Daryl L. Joseffer

CERTIFICATE OF SERVICE

In accordance with Fed. R. App. P. 25 and Fed. Cir. R. 25, this is to certify that I have this day served the foregoing Brief of Appellee via the Court's CM/ECF on all counsel of record.

DATED: July 28, 2014

/s/ Daryl L. Joseffer
Daryl L. Joseffer